College of Computer and Information Sciences

CSC 311 – Design and Analysis of Algorithms — 311 عال –

Midterm I Exam, Spring 2020	Tuesday Feb 25th, 2020	Exam time: 06:00-7:30 P.M.

Student's name: [redacted] ID: [redacted] Section: [illegible]

12 – 11

Problem 1 (3 points)

For each of the question below, insert either T (for **True**) or F (for **False**). **No explanations** are needed. Incorrect answers or unanswered questions are worth zero points.

2.5 / 3

$3000n + 36 = O(n)$	T	
$n \log n + n/5 = O(n^2)$	T	
$n \log n + n^2 = O(n \log n)$	f	
$n^2 \log (n^2) + n^2 = O(n^2)$	f	
$n^2 \log (n-1000) + n^2 = O(n^2)$	f	
The running time of the merge sort is n^2	f	
The lower bound on comparison-based sorting techniques is n	f.	
The time complexity of the **binary search** algorithm is $\Theta(\log n)$	f	X
The worst case running time of insertion sort occurs when the array is sorted.	T	X

Problem 2 (4 points) $\boxed{4}$

Use the formal definition to prove or disprove each of the following. Show ALL steps.

$16n^2 \geq 10n^2 + 6n^2 \geq 10n^2 + 6 \geq f(n) \geq cn^3$

$16n^2 \geq cn^3 \Rightarrow \frac{16}{c} \geq n$

then $f(n) \neq \Omega(n^3)$

2

(a) $f(n) = 10n^2 - 12n + 6 = \Omega(n^3)$.

$5n^2 \geq 2n$

$5n \geq 2$

$n \geq \frac{2}{5}$

$5n^2 \geq cn^3$

$\frac{5}{c} \geq n$

(b) $n(4n + 6) = \Theta(n^2 \log_2 n)$.

$f(n) = 4n^2 + 6n$

$f(n) \neq \Omega(n^2 \log n)$

$10n^2 \geq 4n^2 + 6n^2 \geq f(n) \geq cn^2 \log n$

$10n^2 \geq cn^2 \log n$

$10 \geq c \lg n$

$\frac{10}{c} \geq \lg n$

1

Algorithms	Binary search	Mergesort	Quicksort
Recurrence relation	$T(n) = 2T(\frac{n}{2})$ X	$T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + \Theta(n)$	$T(n) = T(1) + T(n-1) + \Theta(n)$
Running time in big-Oh	$O(n^2)$ X	$O(n \lg n)$ ✓	$O(n^2)$ ✓
Best case running time? Explain.	array come in order X	if array come order. ex) 1, 2, 3, .. X	if take partition random or in medual.
Disadvantage/limitation	if the array come in order ✓	more space in memory ✓	if his take lower number in Array Partition in ✓

Problem 4 (5 points) 4.5

Consider the pseudo-code below:

```
int mystery(int n) {
  int answer;              -> 1
  if (n > 0) {
     answer =(mystery(n/2)+2*mystery(n/4) + n);   T(n/2) + T(n/4) + ..
     return answer;        -> 1
  }
  else
     return 1;             -> 0
```

a- Write the time function, T(n), of this algorithm. Explain. 1

$T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + 2$, the base step when $n = 0$

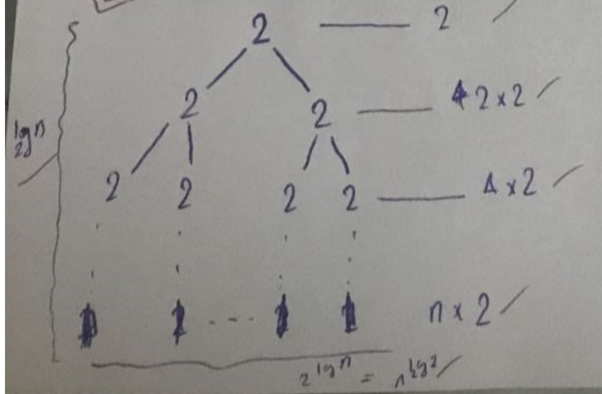
we cont the declration of answer by cost 1, and we go throw if statment we say that method recall it's silf by $\frac{n}{2}$ in first time and the secnd $\frac{n}{4}$)....

$\le \sum_{i=0}^{\lg n - 1} 2^{i} \times 2 + \Theta(n^{\log_2 2}) = (2^{\lg n} - 1) + \cancel{\Theta(n)}\ 2n$

$= n - 1 + \cancel{\Theta(n)}\ 2n$

$= \cancel{\Theta(n)}\ 3n - 1$

2 b- Solve T(n) using the recursion tree.



2

$= 2n (\lg n - 1) + \Theta(n)$

$= 2n \lg n - 2n + \Theta(n)$

$T(n) = 2n \lg n = n \lg n$

we find Guss: $T(n) = n \lg n$

c) induct ~~ract~~ hipatisos

$T(m) \le dm \lg m \quad , \forall m < n, \ d > 0$

$T(n) \le d\frac{m}{2} \lg \frac{m}{2} + d\frac{m}{4} \lg \frac{m}{4} + 2 \le d\frac{m}{2} \lg m - \frac{dm}{2} + \frac{dm}{4} \lg m - \frac{dm}{2} + 2$

$\le d\frac{m}{2} \lg m + d\frac{m}{4} \lg m + 2 \le d\frac{m}{2} \lg m + d\frac{m}{4} \lg m + \lg m$

$\le dm \lg m + \frac{dm}{4} \lg m \le dm \lg m + dm \lg m \le 2dm \lg m$

incorrect.

must be exactly the same as I.H.

we find the Goal $T(n) = dm \lg m$

Base case:

$T(0) = 2$,

$d = 2$

~~$T(n) \cdot T(\frac{n}{2}) + T(\frac{n}{4}) + 2 \quad d > 0$~~

2 , $n = 0$

$\boxed{1.5}$

c- What is the time complexity (*big-O*) of this solution? Prove it.

~~we Guss is $O(n)$ by Q.b~~

~~induct hipatisis:~~

~~$T(m) \le dm \quad , d > 0, \forall m < n$~~

~~the Goal: $T(n) \le dm$~~

~~then $T(n) \le d\frac{n}{2} + d\frac{m}{4} + 2$~~

~~$\le dm + dm + 2 \le 2dm + 2 \le 3dm$~~

~~So we find our Goal~~

~~$T(0) = T(0) + T(0) + 2$~~

~~$= 2 + 2 + 2 = 6$~~

~~Base case:~~

~~$T(0) = 2$~~

~~$d \ge 2$.~~

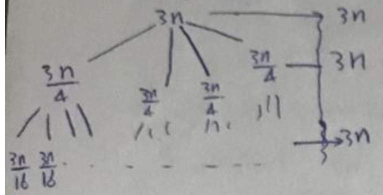
Problem 5 (3 points) $\boxed{2.5}$

Consider the following recurrence relation:

$$T(n) = 4T(\tfrac{n}{4}) + 3n. \rightarrow \text{not tree!!}$$

Solve this recurrence relation using recursive substitutions. Find g(n), where **T(n) = O(g(n))**.

recurrence tree



$$\sum_{i=0}^{\lg n} 3n = 3n\lg n$$

we find the Guss: $O(n\lg n)$

the Goal:

$$\boxed{T(n) \le dn\lg n}$$

inducte hipotisis

$T(m) \le dm\lg m$, $m < n$, $d > 0$

$T(n) \le 4d\frac{n}{4}\lg\frac{n}{4} + 3n$

$\le dn\lg\frac{n}{4} + 3n \le dn(\lg n - \lg 4) + 3n$

$\le dn\lg n - 2dn + 3n \longrightarrow +2dn \ge 3n$ $\boxed{d \ge \frac{3}{2}}$

we Pick $d > n \longrightarrow$ illegal

$\le dn\lg n$

$T(n) = O(n\lg n)$

Problem 6 (2 points)

S A 4 4 S A A d

1

m S A A d

Given a string S of n characters and an integer $k < n$, does the string S contain any duplicate substrings of size k?

For example, with S = "ABABAB" and $k = 2$, the answer is *true*, since the substring "AB" appears many times. For S, and k=4, the answer is also yes, because "ABAB" appears twice. For k=5, the answer is *false*.

- Design a brute-force algorithm to solve this problem.

```
alg: dupSup
X input: two array of one char and secnd string, A, B of size k, n
output: true or false

for i <- 1 to n-k+1 do
    m <- 0, L <- i
    while A[L] = B[m] & L > m
        m++, L++
    if m = k
        return true
return false
```